Amendments to the Claims

This listing of claims will replace all prior versions and listings of claims in the application:

Listing of claims:

1. (currently amended) A method of generating a key *k* for use in a cryptographic function performed over a group of order q, said method including the steps of:

generating a seed value *SV* from a random number generator;

performing a hash function *H()* on said seed value *SV* to provide an output *H(SV)*;

determining whether said output *H(SV)* is less than said order q;

accepting said output *H(SV)* for use as [[a]] said key *k* if the value ~~thereof~~ of said output *H(SV)* is less than said order q; [[and]]

rejecting said output *H(SV)* as [[a]] said key if said value is not less than said order q[[.]];

if said output *H(SV)* is rejected, repeating said method; and

if said output *H(SV)* is accepted, providing said key *k* for use in performing said cryptographic function, wherein said key *k* is equal to said output *H(SV)*.

2. (original) The method of claim 1 wherein another seed value is generated by said random number generator if said output is rejected.

3. (original) The method of claim 1 wherein the step of accepting said output as a key includes a further step of storing said key.

4. (original) The method of claim 1 wherein said key is used for generation of a public key.

5. (previously presented) The method of claim 1 wherein said order q is a prime number represented by a bit string of predetermined length L.

6. (previously presented) The method of claim 5 wherein said output from said hash function is a bit string of predetermined length L.

7. (original) The method of claim 1 wherein if said output is rejected, said output is incremented

by a deterministic function and a hash function is performed on said incremented output to produce a new output; a determination being made as to whether said new output is acceptable as a key.

8. (original) The method of claim 7, wherein said step of incrementing includes a further step of adding a particular value to said seed value.

9. (currently amended) A method of generating a key *k* for use in a cryptographic function performed over a group of order q, said method including the steps of:

generating a seed value *SV* from a random number generator;

performing a hash function *H()* on said seed number *SV* to provide a first output *H(SV)*;

incrementing said seed value *SV* by a predetermined function *f()* and performing said hash function *H()* on said incremented seed value to provide a second output *H(f(SV))*;

combining said first output *H(SV)* and second output *H(f(SV))* to produce a new output;

determining whether said new output has a value less than said order q;

accepting said new output for use as [[a]] said key [[k]] *k* if said new output has a value less than said order q; [[and]]

rejecting said new output as [[a]] said key *k* if said ~~new output has a~~ value is not less than said order q[[.]]

if said new output is rejected, repeating said method, and

if said new output is accepted, providing said key *k* for use in performing said cryptographic function, wherein said key *k* is equal to said new output.

10. (original) The method of claim 9 wherein upon rejection of said new output a new seed value is generated by said random number generator.

11. (original) The method of claim 9 wherein upon rejection of said new output said seed value is incremented by a predetermined function and revised values for said first output and said second output are obtained.

12. (previously presented) The method of claim 9 wherein a bit string greater than a predetermined length L is obtained and an L bit string selected therefrom for comparison with said order q.

13. (previously presented) The method of claim 12 wherein upon rejection of said bit string of predetermined length L, a further L bit string is selected.

14. (previously presented) The method of claim 9 wherein said step of combining said first and second outputs includes a further step of rejecting excess bits such that said new output is a bit string of length L.